



Texas Instruments Software

ENABLING SOLUTIONS WITH DSDC & TI PARTNERSHIP

Presented By: Joseph A. Ghattas
Government Solutions
jagz@mimi.ti.com



Agenda

- Comprehensive Solution
- Paternship for Solution Delivery
- Pathfinder Approach
- Open Discussion



Comprehensive Solution

- Infrastructure
 - Strategic Planning and Implementation Road Map
 - Development Coordination
- Process
 - Business Process Engineering
 - Development Methodologies
 - Transition Engineering
- People / Resources
- Tools



Enterprise Information Architecture

- Encyclopedia Management
- Development Coordination

Information Infrastructure

- Information Engineering

Improve Processes

- BPE

Build Systems

- Components
- Transition Engineering

Exploit Systems

- System Deployment
- COTS/GOTS Integration

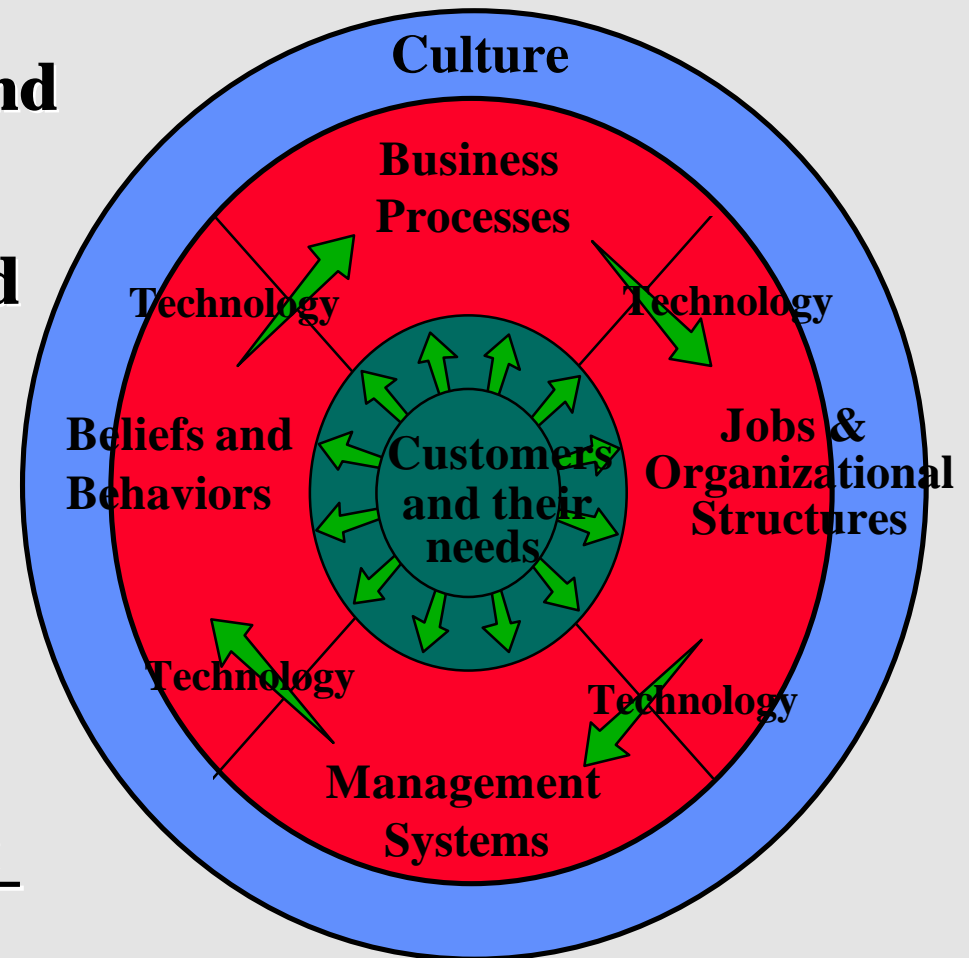


Business Process Engineering

**Fundamental analysis and
radical redesign of
business processes and
related areas.**

**Align all resources to
meet the needs of the
customer.**

**Success requires a holistic
approach.**



Adapted from
Dr. Michael Hammer's Reengineering Diamond

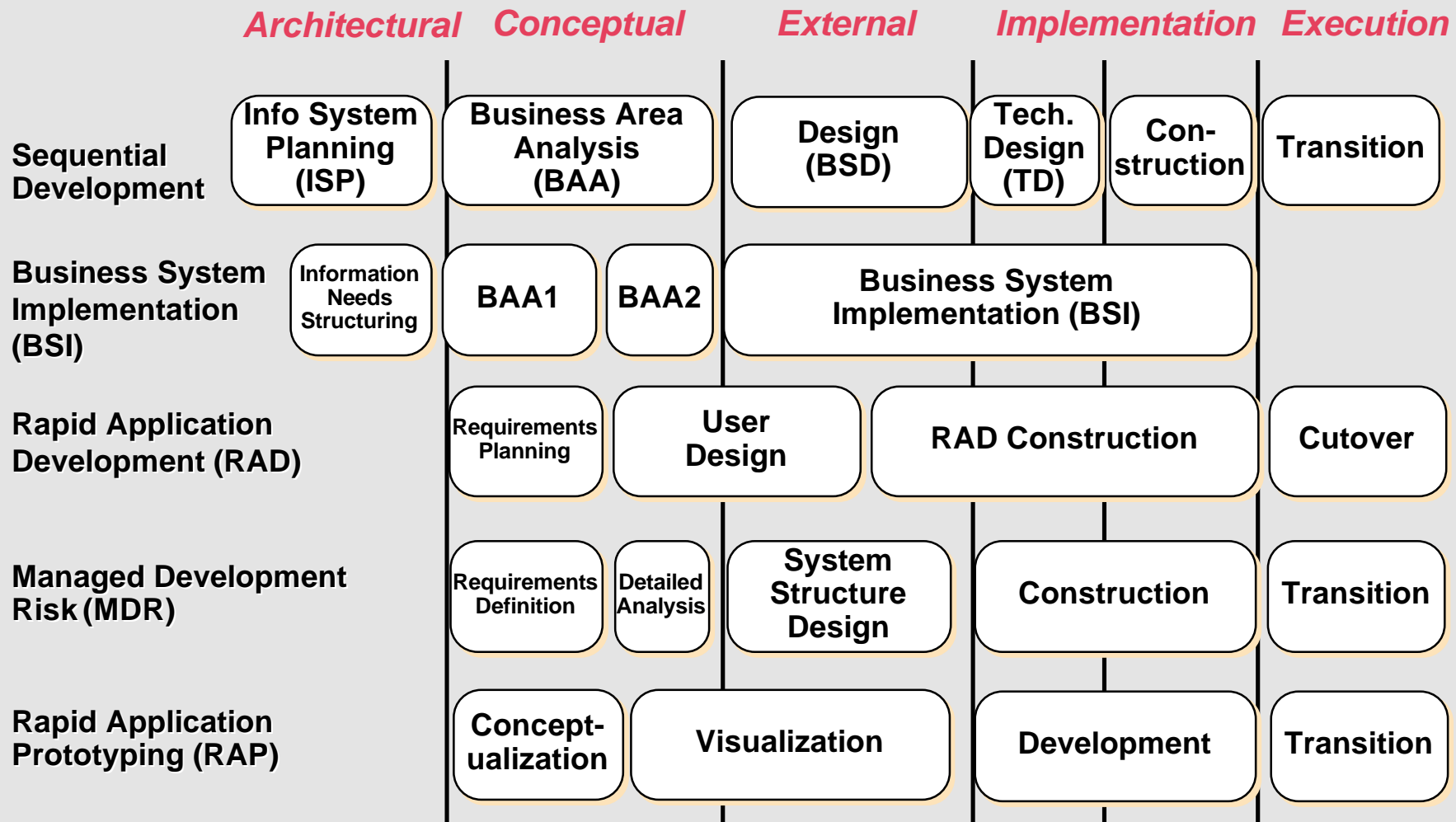


Composer Approach

- **Full Life Cycle**
 - Development and maintenance
- **Model Driven**
 - Focuses at business level
 - Consistent objects, terminologies, and Methodologies
- **Platform Independent**
 - Model is developed independent of target environment
 - Multiple environments supported for development toolsets and Application targets.
- **Scalable**
 - Easy to change target environments (e.g., Windows 3.1, HP/UX, MVS, stand alone, Client/Server)
- **Fully Integrated**



Alternative Development Life Cycles



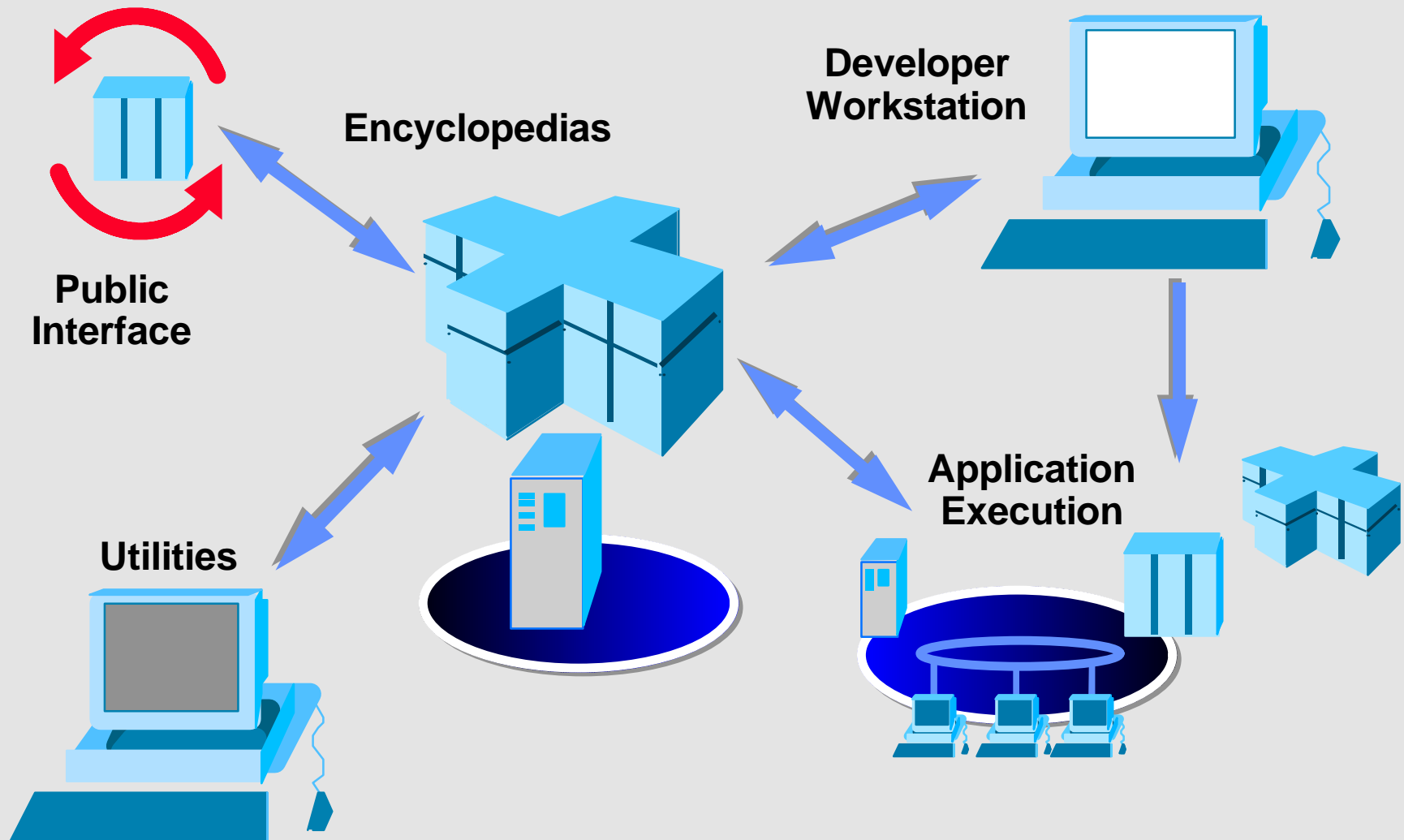


Composer Architecture

- Workstation Toolsets
 - Used for model development
 - Planning, Analysis, Design, Construction (100% code generation)
 - Seamless Connectivity to encyclopedias
- Encyclopedias
 - Model repository
 - Allows concurrent development activities on a model
 - Two types
 - Central Encyclopedia (CE) - MVS Host
 - Client/Server Encyclopedia (CSE) - Midtier (e.g., HP/UX)
- Implementation Toolset (IT)
 - Builds application executables for a target environment



Composer Development Environment

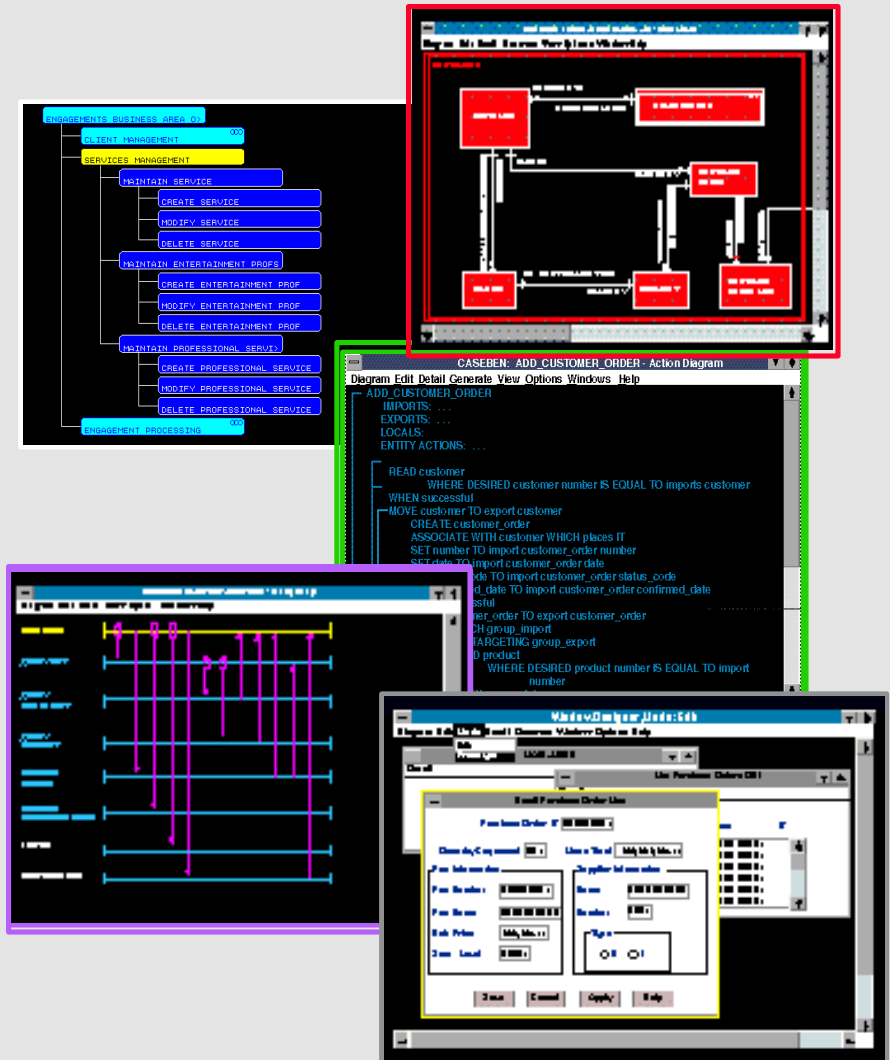




Model-Driven Development

Accurately reflects the business

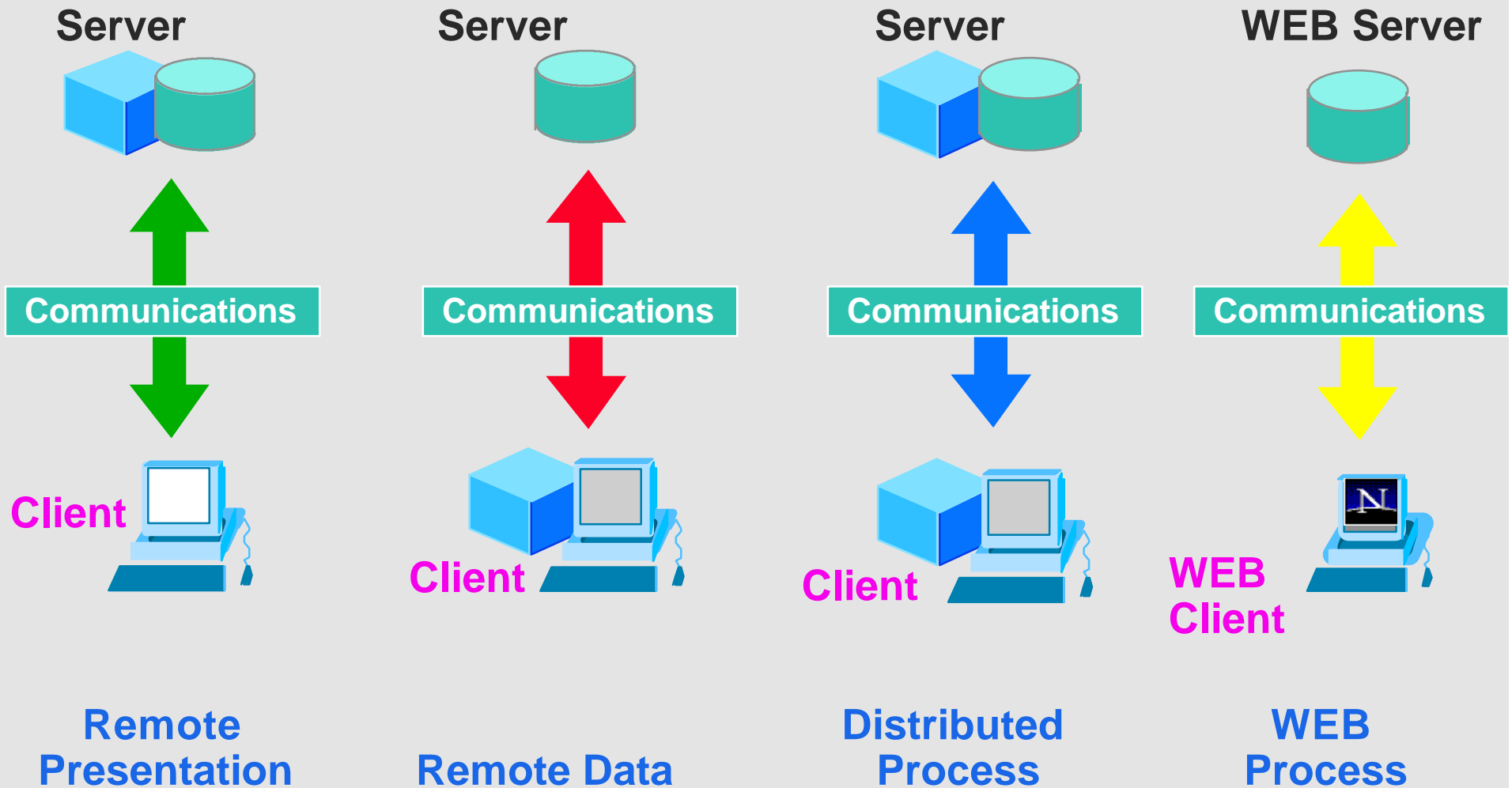
- Quickly respond to change
- Conceptualize complex problems
- Consider business alternatives without constraints of technology
- Consistently build, install, test, modify and enhance the business application



Composer Diagrams - Fully Integrated Model



Client / Server Styles





Composer Summary

- **Model-Driven Development and Maintenance**
- **Fully Distributed Applications**
- **100% Generation of Application**
 - **Client**
 - **Server**
 - **Database**
 - **Communications Protocol**
- **Advanced GUI Design and Construction**
- **Platform & Database Flexibility**
- **Extended Communications Infrastructure - Pipes**
- **Scalable Open Architecture**
- **Internet Client Access**



Texas Instruments Software

Full Lifecycle Support

TI Common Architecture

Conceptual

Deployment

Disciplines

BPR

Data
Administration

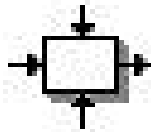
Standards

Software Process
Improvement

Software
Reuse

Infrastructure

Methods



IDEF

Version
Control

MIL-STD-498
DoD 8020
DoD 8320

IE

Object &
Component Reuse

Multiple Platforms
OSE - TAFIM

SEI
CMM

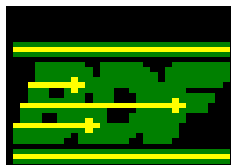
GUI / CS

Communication
Protocols;
PIPES

Model
Management

Transition Engineering

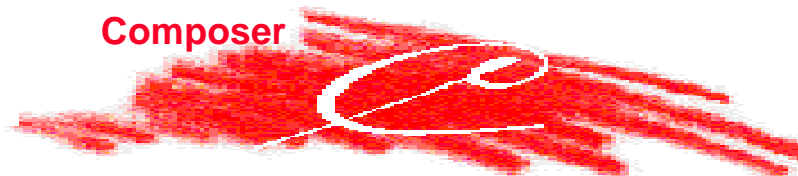
Tools



Central Encyclopedia / Client/Server Encyclopedia

Composer

arranger





Transition Solutions Process

“Create flexible, responsive business applications from troublesome legacy systems”

Focus:

Derive business model from legacy system

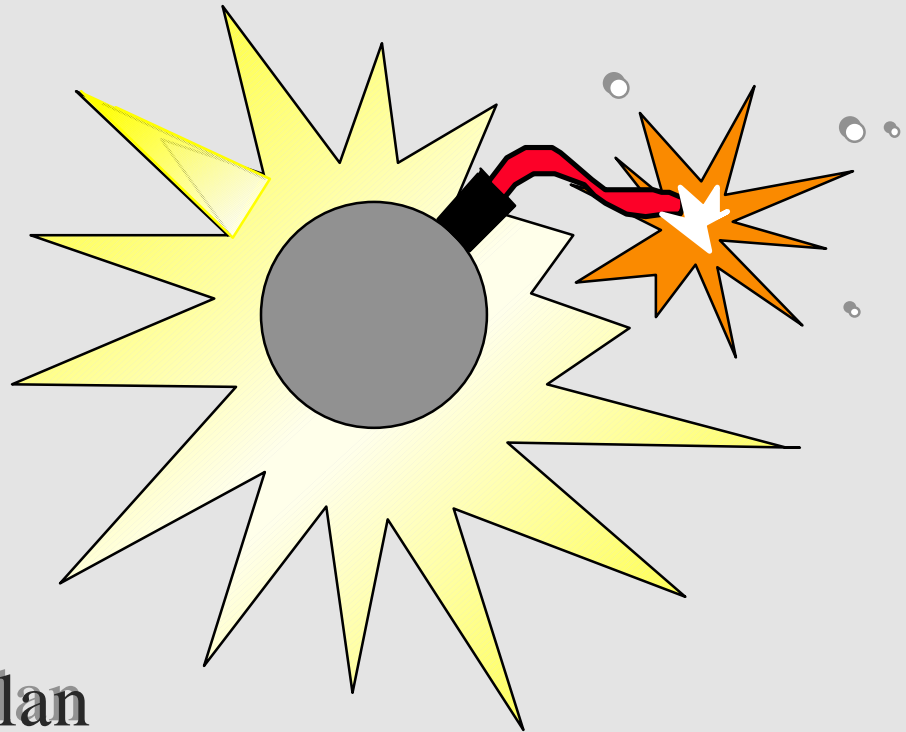
Extract business rules

Derive value from reuse of existing components



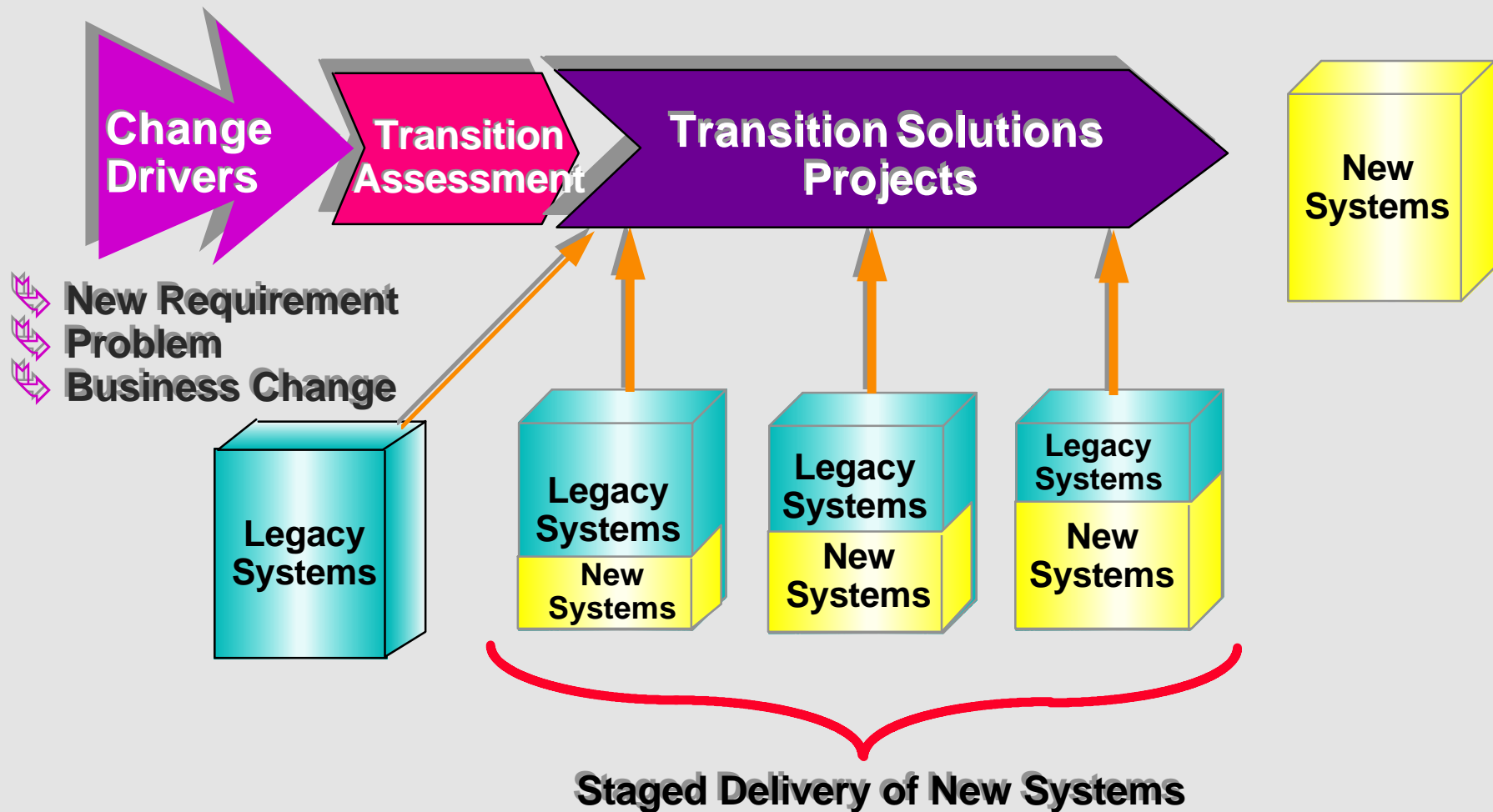
Problems with the Typical Response

- “Big Bang”
 - Risk
 - Cost
 - Time
- Survival of Current system
- No contingency plan



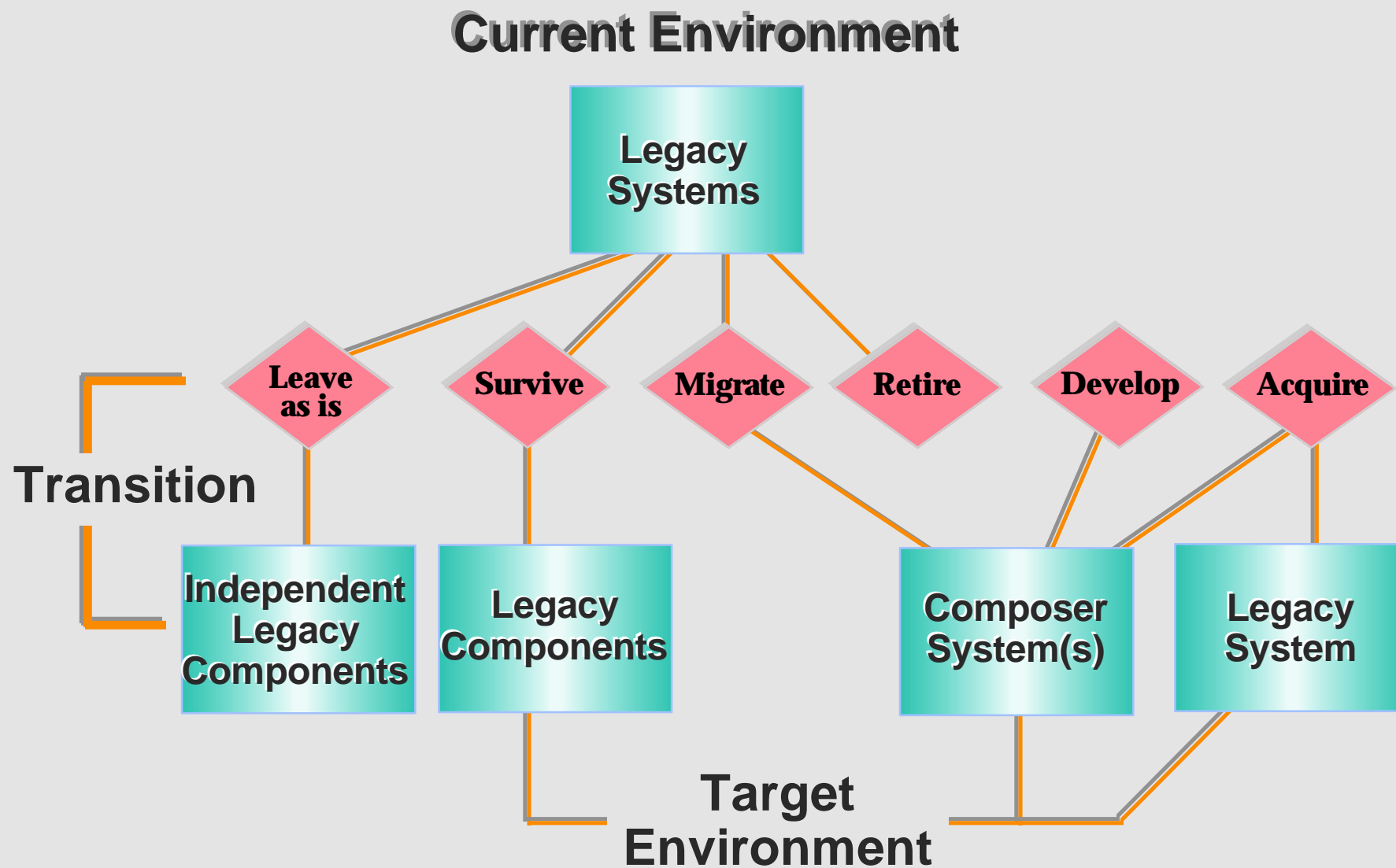


Incremental and Orderly Transition in Response to Change





Transition Scenarios





Transition Engineering Solution

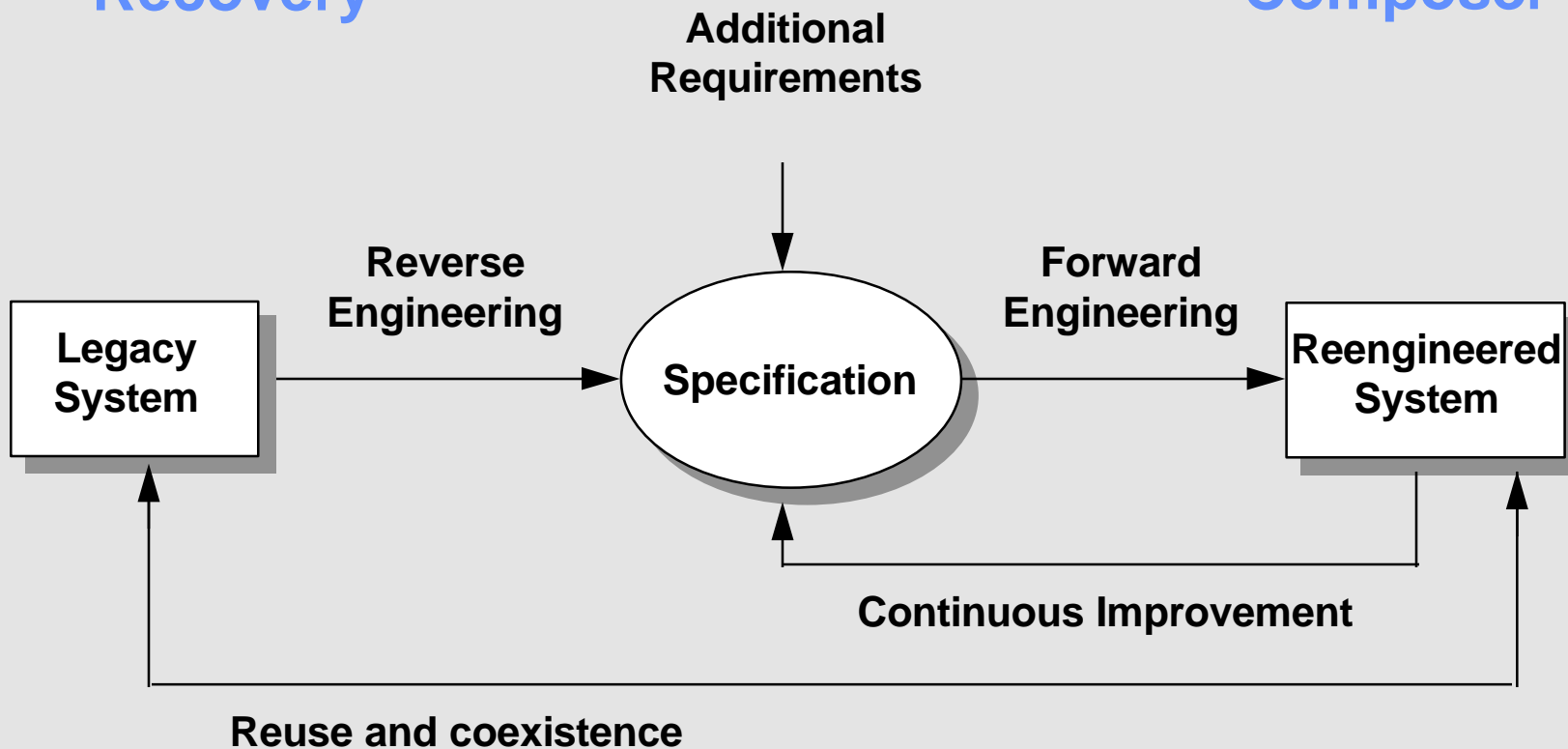
- **Planning** -- Inventory, Scope & Understand Legacy Systems to Develop Transition Approach
- **Analysis** -- Identify & Recover Legacy Components for Process and Data Models & Business Rule Tracing
- **Implementation** -- Release Management Activities to Implement Changes to Legacy System Components



The Transition Solutions Process

Automation of
Recovery

Integration with
Composer



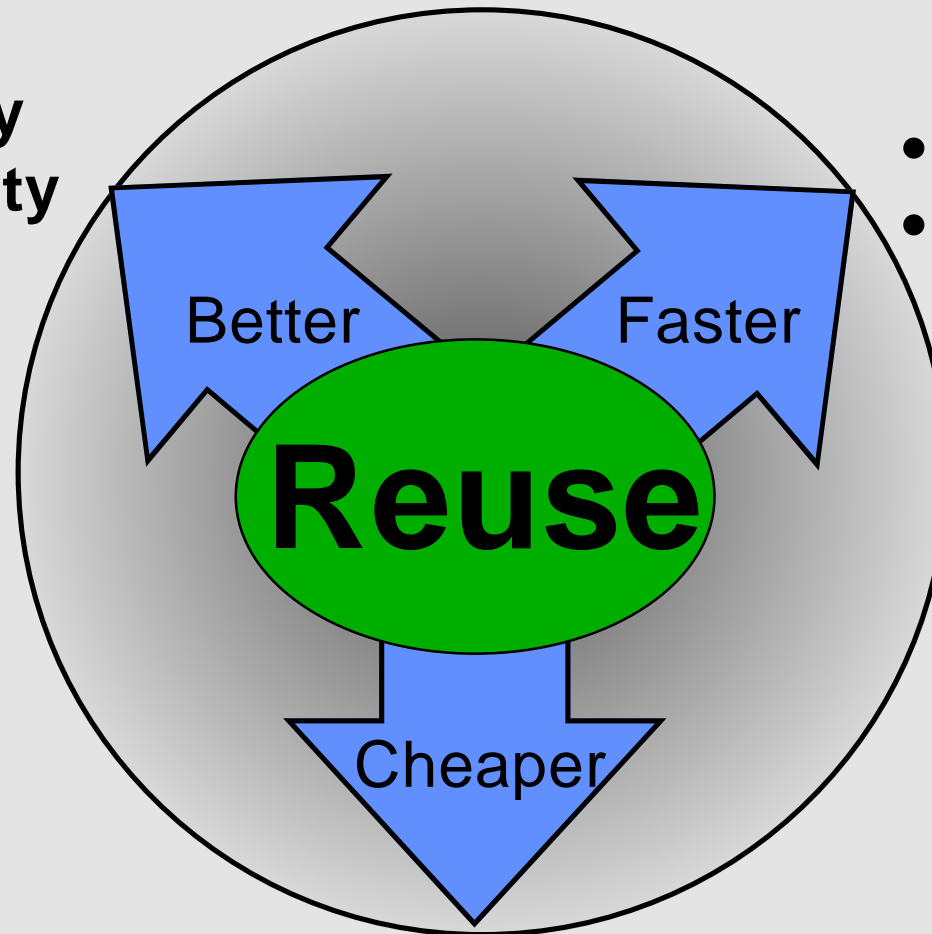
Legacy Coexistence

Incremental delivery



Business Demand for Systems

- **Quality**
- **Integrity**

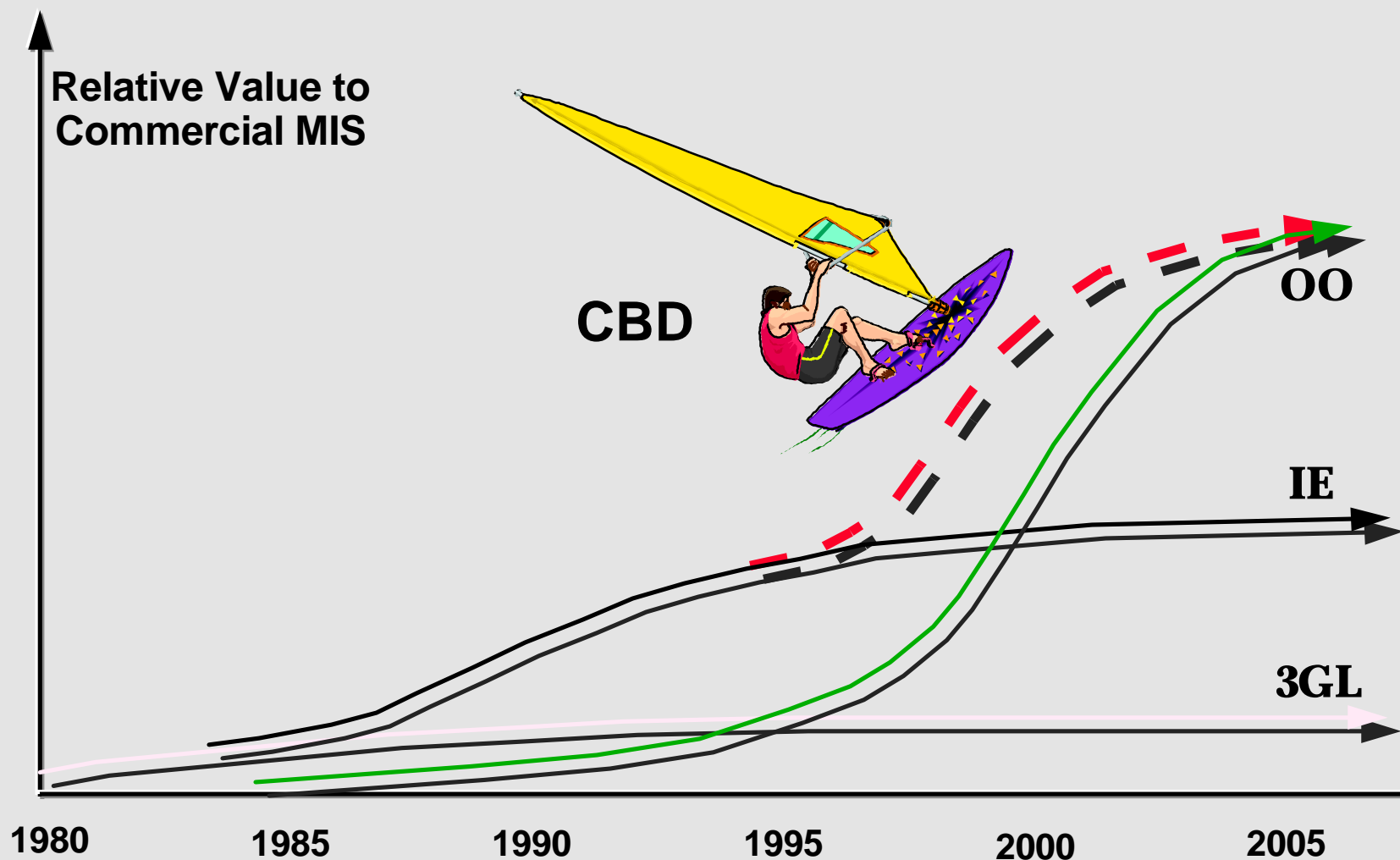


- **Cycle Time**
- **Flexibility**

- **Cost of Ownership**
- **Best of Breed**



CBD ***Getting Ahead of the Curve***





Component-Based Development

- Radical Development Time Reduction
- Plug and Play
- Extensive Reuse
- Enablement of OO techniques



So, What is a Component?

An independently deliverable package of defined services

■ **Every Component has:**

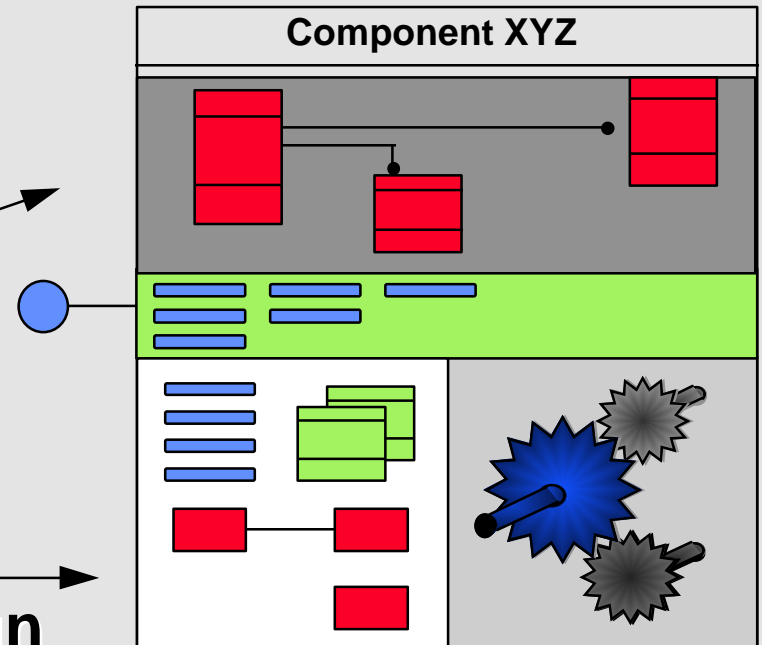
■ **A specification**

■ **An interface**

■ **An implementation design**

■ **Executable software module(s)**

■ **Not all have to be delivered together to have value**





IE Market Characteristics

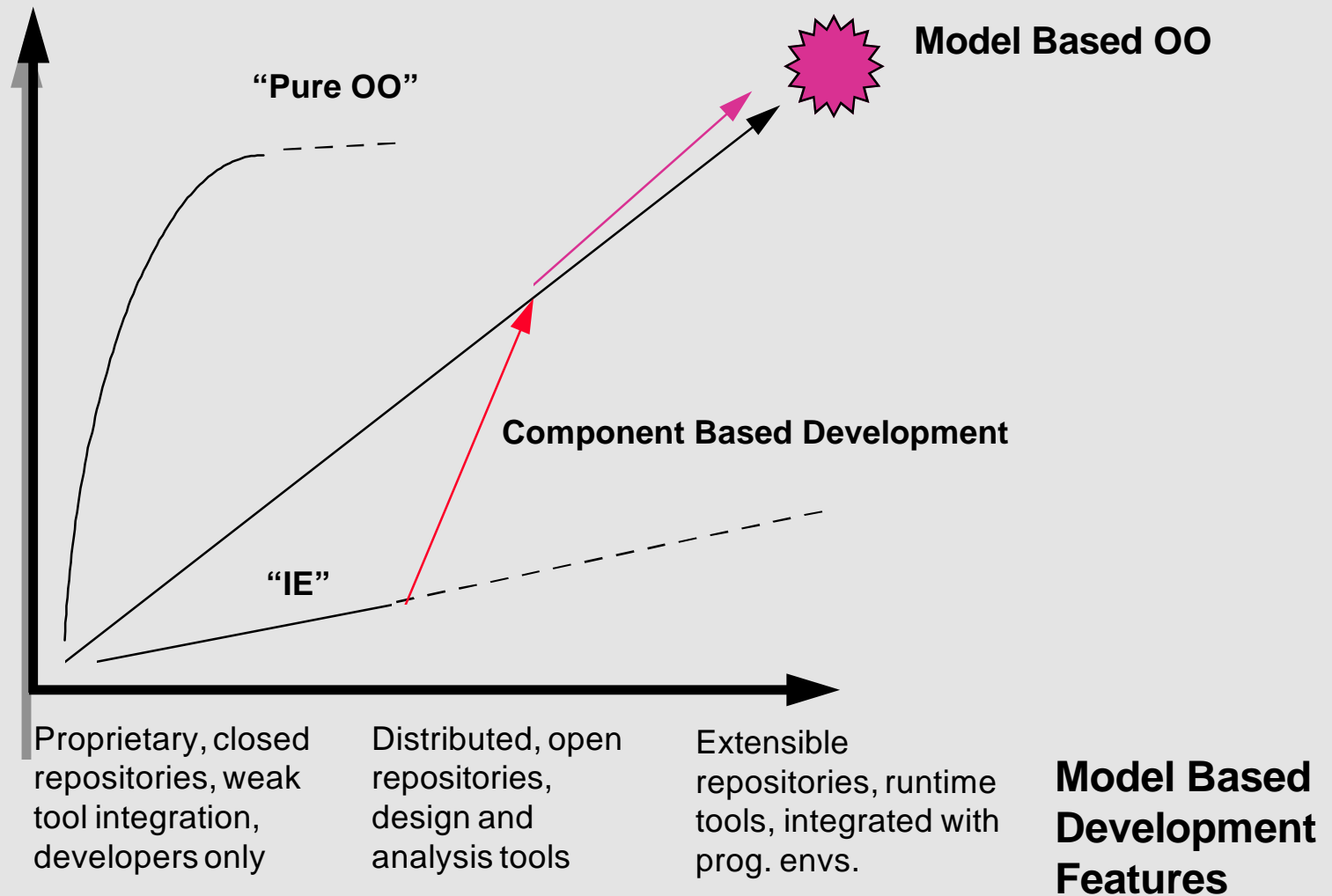
OO Features

Generic Types

Polymorphic
Behaviour,
Inheritance

Encapsulation
& Interfaces

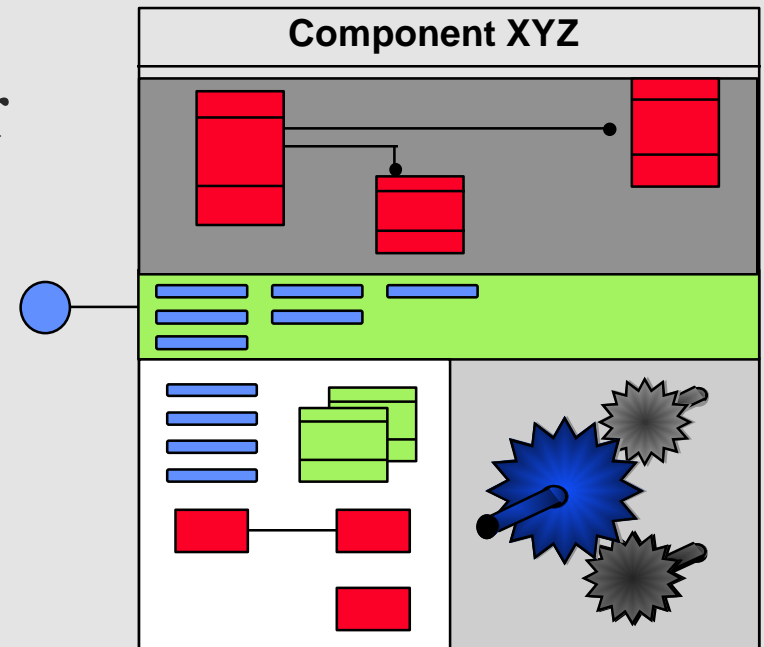
Modular
Programming





The Opportunity!

- Exploit existing software components with Arranger
- Embrace the Component Based Development Architecture
- Establish an internal Software Factory





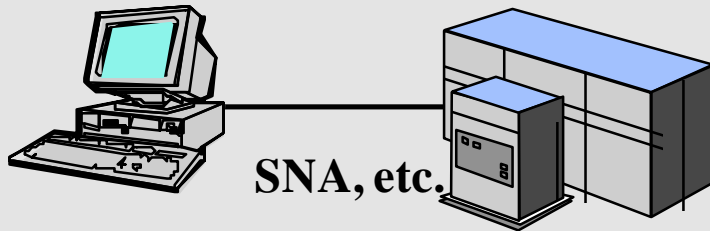
Key Component Methodology Features

- Builds on Existing OO and IE Methods
- Automatable
- Strong Component Orientation
- Formal Refinement Process
- Rigorous Component Protocol Modeling
- Supports OO and Non-OO Development Environment



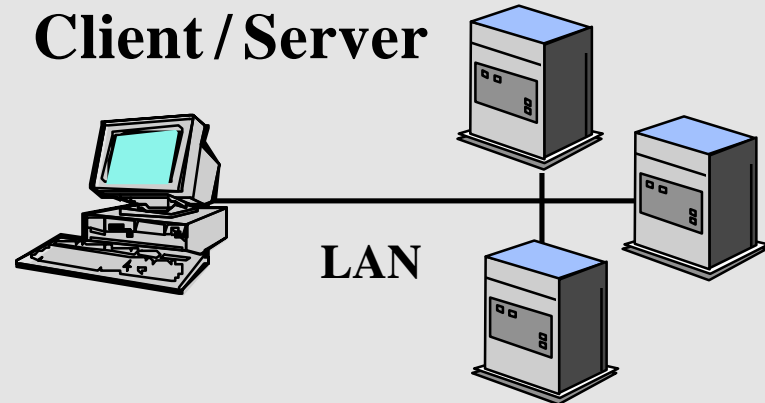
Application Development FUTURE PROOFING

Terminal To Host



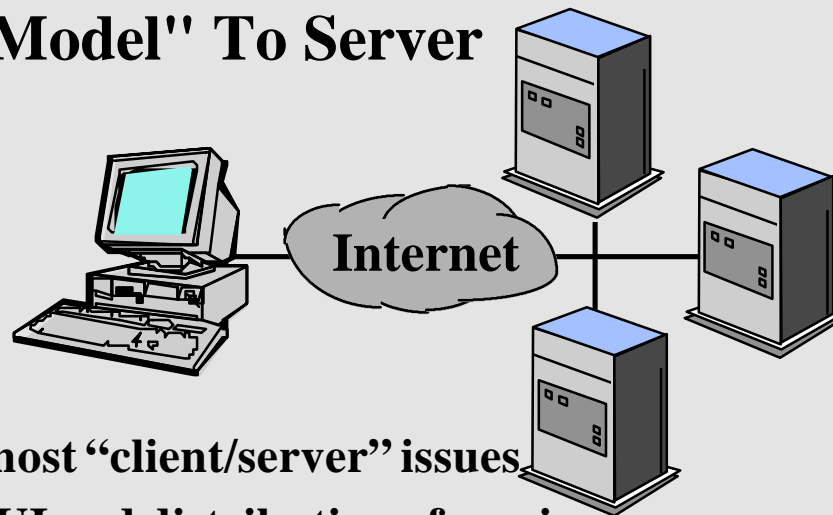
**Terminal “owned” by host
All “instructions” from host**

Client / Server



**Workstation independent of server
Client code not independent of server code**

"Web-Model" To Server



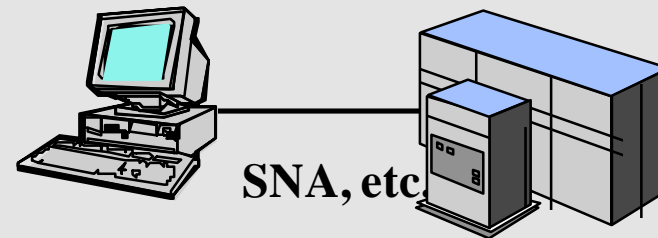
**Resolves most “client/server” issues
Retains GUI and distribution of services**



Architecture Evolution In Progress

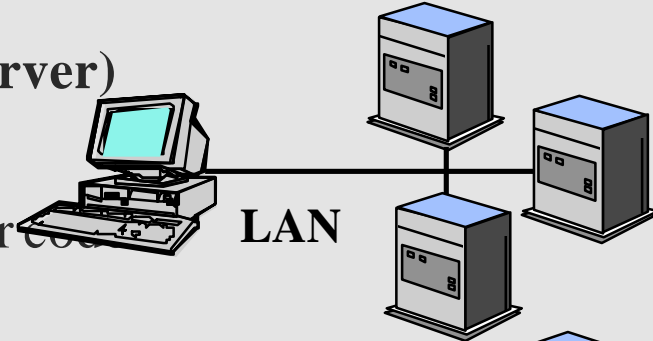
■ Terminal to host

- Terminal “owned” by host
- All “instructions” from host
- Must go “through host” to access other hosts



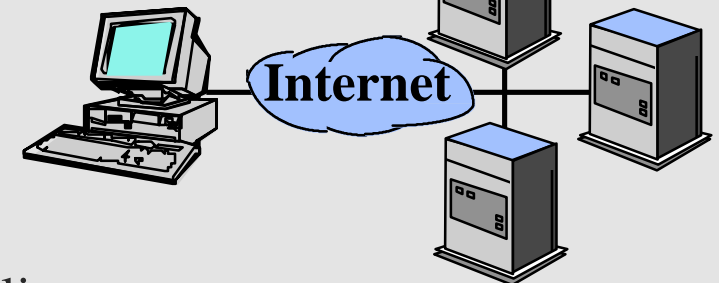
■ Workstation to server (Client / Server)

- Workstation independent of server
- Client code not independent of server code
- Software distribution a major issue
- Client environment a major issue

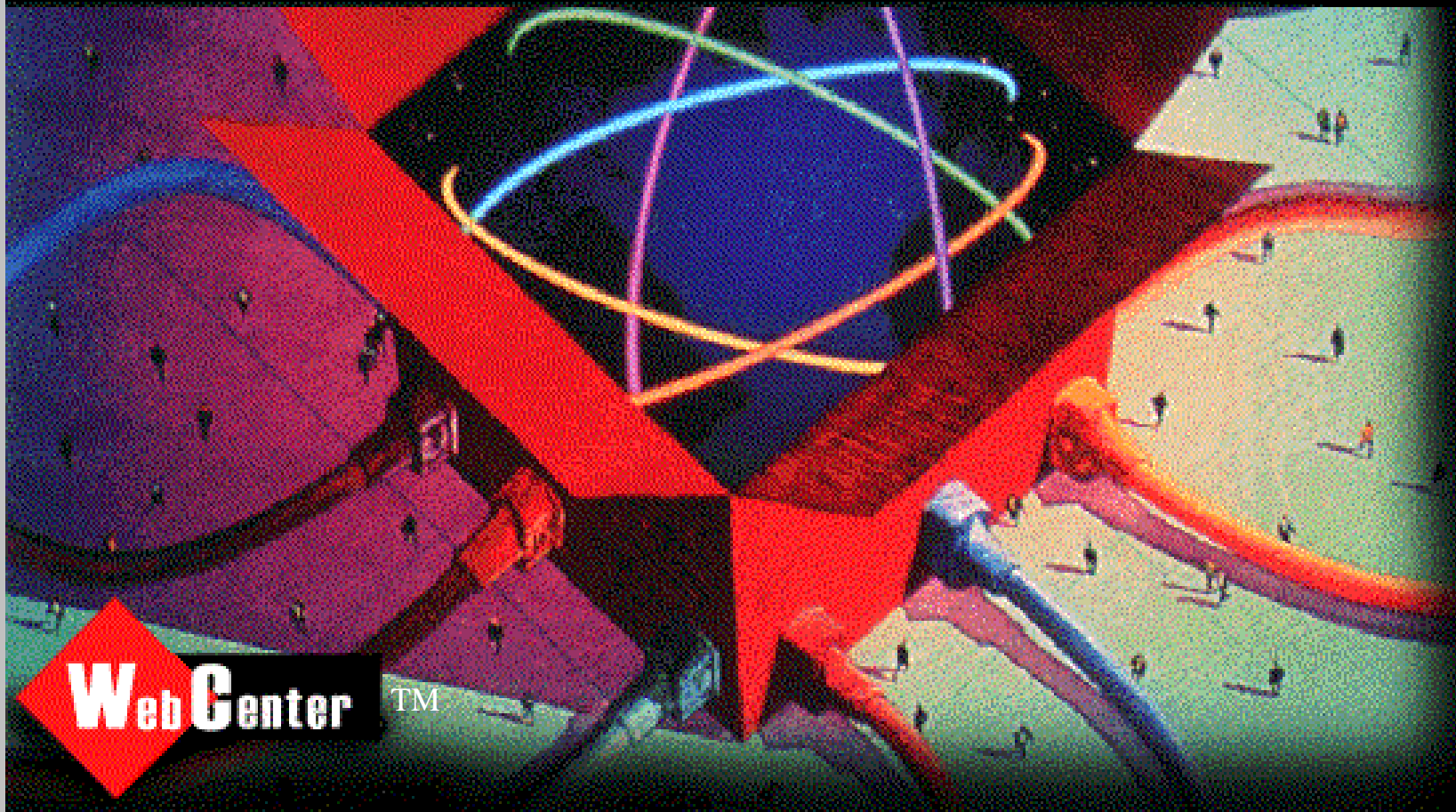


■ “Web-model” to server

- Resolves most “client/server” issues
- Retains GUI and distribution of services
- Application writer not concerned about client
- Intimate client/server development evolves to anonymous
- Client independent from host/server



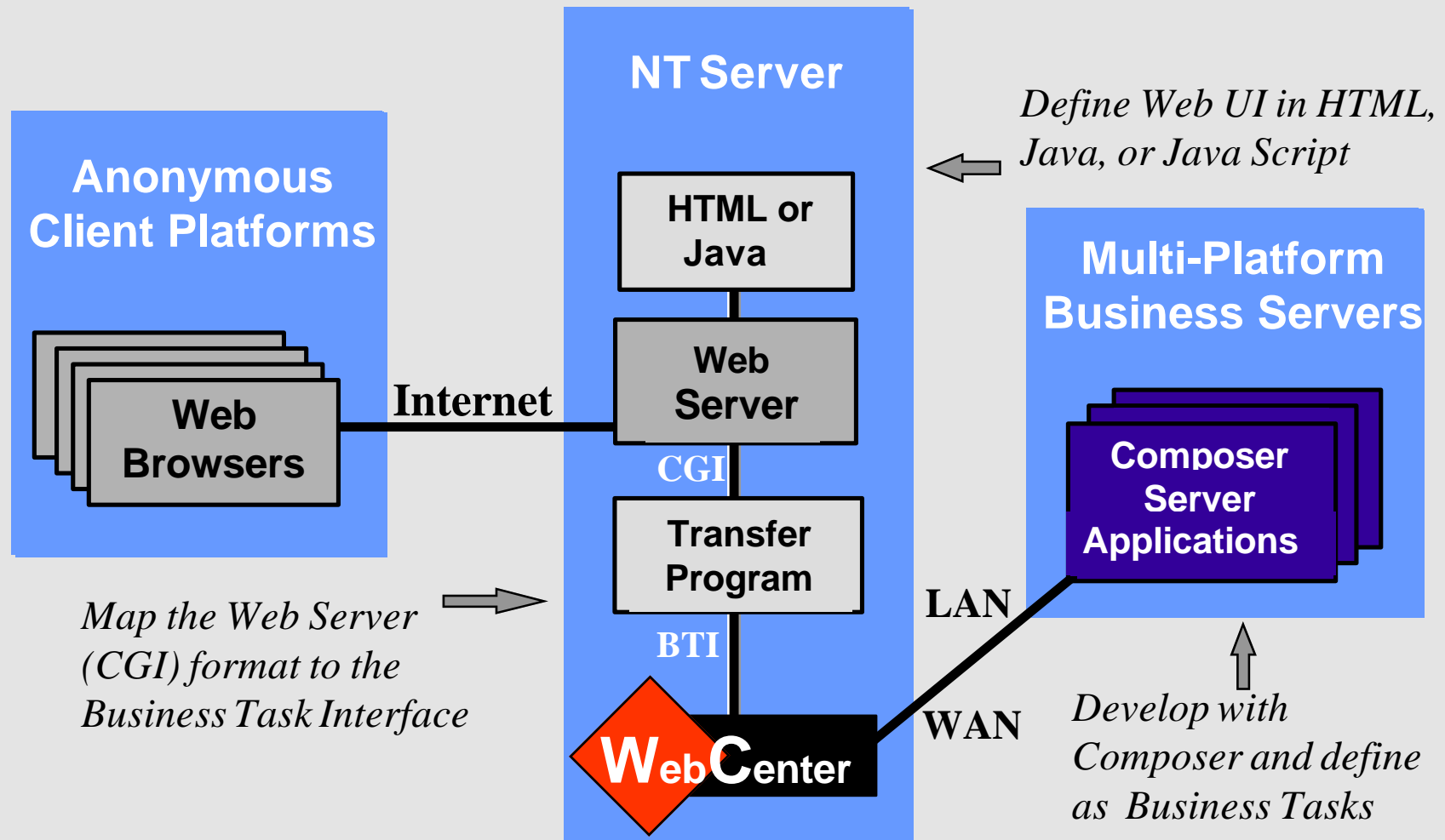
 **TEXAS INSTRUMENTS**



Internet for the EnterpriseTM



3 Steps To The Internet





TI Services: Objectives

- Technology Transfer
- Ultimate Source of COMPOSER Technical Expertise
- Successful System Deployment
- Customer Self-Sufficiency



TI Services :Training

- COMPOSER Client Server Development Principles
- C/S Encyclopedia Overview & Subsetting
- Host Encyclopedia: Version Control
- Rapid Skills Development Workshop
- Building GUI Applications
- COMPOSER Project Management
- COMPOSER Development Coordination
- BPE Practitioner's Workshop



TI Consulting Services

- Project Planning & Scoping
- Project Assessment
- Business Process Engineering
- BAA Facilitation/Requirements Gathering
- Client Server Design & Development
- COMPOSER Rollout/Implementation Planning
- COMPOSER Development Coordination
- Legacy System Reengineering



Development Coordination

- Information Resource Administration
- Architecture Management
- Development Planning
- Project Management
- Encyclopedia Administration
- Model Management
- Version Control
- Configuration Management
- Change Control
- Release Management



Composer: How to Get Started

- Enterprise Rollout: Roadmap Project
 - ◆ Management of Implementation
 - ◆ Organizational Learning
 - ◆ Tools & Methods
 - ◆ Architectures & Strategies
 - ◆ Databases & Applications
- Pathfinder Project: 3 - 5 Months
 - ◆ Just-in-Time Training
- Benchmark: 1 - 2 Weeks



Pathfinder Projects

- 3 to 5 Month Duration
- 10 to 12 Team Members
- Managed to Success
- Full Lifecycle Education
- Full Time Consulting Support
- Key Objectives:
 - Demonstratable/Implementable Results
 - Establish Internal Expertise
 - Develop Initial Infrastructure and Standards



Elements of Project Success

- Management Commitment
- Well-Defined Project / Scope & SOW
- Project Schedule Aggressive, But Achievable; Progress Monitored; Early Results
- Team Members Experienced, Trained & Motivated; Just In-Time Training
- Users/Owners of System are Part of Team & Able to Make Decisions On-the-Spot
- Incremental & Continuous Delivery -- Success is Contagious

Sound Implementation Planning